【別紙】アーキテクチャ比較表

No.	アーキテクチャ	説明	PostgreSQL	Oracle	SQL Server	DB2
1	プロセス構成	DBサーバのプロセスの構成	マルチプロセス構成。マスタープロセスがリスナプロセスであり、クライアントからの接続を受け付ける毎にバックエンドプロセスが生成される postgres常駐プロセス(postmaster) [*1] postgres子プロセス [*2] 補助プロセス [*3] ライタ WALライタ WALアーカイバ 統計情報コレクタ ロガー 自動バキュームランチャ 自動バキュームワーカ WALセンダ WALレシーバ など [*1]マスタープロセス。バックエンドを管理する常駐プロセス(PostgreSQL管理デーモン) [*2]バックエンドプロセス。フロントエンドからの接続毎に常駐プロセスから生成(fork)されるデータベースエンジン [*3]postgres常駐プロセスから起動されるその他のプロセス群	マルチプロセス/スレッド構成。ネットワーク・リスナー・プロセスが接続を受け付け、サーバープロセスが起動される ネットワーク・リスナー・プロセス [*1] サーバープロセス [*2] ディスパッチャ・プロセス [*3] 共有サーバー・プロセス [*3] 専用サーバー・プロセス [*4] バックグラウンドプロセス [*2] データベース・ライター（DBWR） ログ・ライター（LGWR） チェックポイント（CKPT） システム・モニター（SMON） プロセス・モニター（PMON） リカバラ・プロセス（RECO） アーカイバ・プロセス（ARCn） ジョブ・キュー・プロセス キュー・モニタ・プロセス など [*1]Oracle Net Servicesの一部 [*2]Windows環境ではデータベースインスタンスが1プロセス（バックグラウンドプロセスはスレッドとして実装されている） [*3]共有サーバ接続の場合 [*4]専用サーバ接続の場合	マルチプロセス/スレッド構成。各プロセスはWindowsのサービスとして実装されている SQL Server データベース エンジン [*1] SQL Server エージェント [*2] SQL Server Integration Services [*3] SQL Server Browser [*4] レプリケーション エージェント [*5] SQLライター サービス [*6] その他サービス SQL Server Analysis Services SQL Server Master Data Services SQL Server Reporting Services など [*1]データベースエンジンのインスタンス。Windowsのサービスとして実行される。(sqlservr.exe 実行可能ファイルのコピー) [*2]管理ジョブを自動実行しジョブの実行、SQL Serverの監視および警告の発生を行う [*3]データ転送サービス [*4]SQL Server の各種リソースに関する着信要求を受信し、SQL Server インスタンスに関する情報を提供する [*5]SQL Server エージェントから実行されるレプリケーション関連のスタンドアロンプログラム。スナップショットエージェント、ログリーダーエージェントなど [*6]SQL Server のバックアップと復元に関する追加機能を提供	マルチプロセス/スレッド構成。接続毎にコーディネーター・エージェントが生成されクライアントからの要求を処理する 通信リスナー コーディネーター・エージェント [*1] データベースEDU [*2] メイン・システム・コントローラ バッファー・プール・プリフェッチャー用 バッファー・プール・ページクリーナー用 ログマネージャ用 ログトラッキング用 デッドロック検出用 イベント・モニター高速ライター など [*1]接続毎。データベース要求を実行する [*2]EDU：エンジン・ディスパッチ可能単位 様々なタスクの実行を行う実行単位（プロセス、スレッド）
2	メモリ構成	DBサーバのメモリの構成	大きくはプロセス間で共有されるメモリ領域とプロセスに固有のメモリ領域に分けられる 共有メモリ [*1] 共有バッファ [*2] WALバッファ [*3] フリースペースマップ [*4] ビジビリティマップ [*5] など プロセスメモリ [*6] 作業領域 [*7] メンテナンス領域 [*8] 一時バッファ など [*1]複数のバックエンドで共有されるデータの保持領域。サービス起動時に固定サイズが確保される [*2]テーブルやインデックスのデータをキャッシュする領域 [*3]ディスクに書き込まれていないトランザクションログ（WAL：Write Ahead Logging）をキャッシュする領域 [*4]テーブル上で利用可能な空き領域を指し示す情報を扱う領域 [*5]テーブルのデータが可視であるかを管理する領域 [*6]実行プロセス毎に確保される [*7]クエリ実行時の並び替え、ハッシュテーブル操作のために使われる領域 [*8]インデックス作成、外部キー追加、バキュームなど、メンテナンス操作で使用する領域	大きくはOracleインスタンス内で共有されるメモリ領域（SGA）とプロセスに固有のメモリ領域（PGA）に分けられる。それら内部にセッション単位の領域（UGA）、スタック領域（CGA）が確保される SGA [*1] 共有プール データベース・バッファ・キャッシュ REDOログ・バッファ ラージ・プール Javaプール Streamプール 結果キャッシュ PGA [*2] UGA [*3] CGA [*4] [*1]共有メモリ領域。自動メモリ管理機能使用の場合はメモリファイルシステム [*2]動作中のプロセスやスレッドに固有のメモリ [*3]主にSQLのソートや表結合などの処理で利用される領域。セッション単位で確保される。専用サーバ接続であればPGA、共有サーバ接続であればSGAのLARGEプールの中から確保される [*4]PGA内に確保されるスタック領域	メモリの確保と解放が必要に応じて動的に行われる ■動的メモリ管理 使用可能なシステム リソースに基づいて必要なメモリ量が動的に変更。最小の既定メモリ量は、0MBとなっている。最大メモリ量に設定されている既定値は、2GBで、最大メモリ量に設定できる最小値は、16MBとなっている ■AWE(Address Windowing Extensions) 32ビット版のオペレーションシステムのみに適用され、4GBを超える物理メモリを使用でき、最大64GBの物理メモリをサポートする ■バッファ管理 I/Oの効率向上を実現するために重要なコンポーネントで、データベースページに対するアクセスと更新を行うバッファ マネージャと、データベース ファイルのI/Oを削減するバッファ キャッシュから構成される ■オブジェクトの使用で使用されるメモリ ・テーブル ロック等 ロックしたことによるメモリ消費 ・クライアントが接続したことによるメモリ消費	メモリーセットという管理単位を複数持つ構成 インスタンス・メモリ DB(Database) memory set [*1] Application memory set [*2] DBMS memory set [*3] FCM memory set [*4] FMP memory set [*5] Private memory set [*6] Local Communication memory set [*7] [*1]データベース毎。各データベース固有の処理用 ・バッファー・プール・ヒープ ・パッケージ・キャッシュ・ヒープ ・カタログ・キャッシュ・ヒープ ・データベース・ヒープ ・ロック・マネージャ・ヒープ　など [*2]データベース毎。アプリケーション固有の処理用（作業領域等） ・アプリケーション・ヒープ ・ステートメント・ヒープ ・統計ヒープ　など [*3]インスタンス毎。通信サービスなどの基本インフラストラクチャーの目的で使用 [*4]ホスト毎。各パーティション間でのFCMによる通信に使用 [*5]インスタンス毎。エージェントとfenced モード・プロセス(db2fmp)の間の通信に使用 [*6]インスタンス毎。内部管理領域 [*7]接続毎。ローカル・アプリケーションとその関連エージェントの間の通信に使用

【別紙】アーキテクチャ比較表

No.	アーキテクチャ	説明	PostgreSQL	Oracle	SQL Server	DB2
3	ディスク構成	DBサーバのディスクに配置される物理ファイルの構成	インスタンス（データベースクラスタ）は基本的には1つのディレクトリで管理される データベース毎の情報はサブディレクトリ内にファイルとして格納される [データベースクラスタディレクトリ] +- [データベースディレクトリ] [*1] | +- テーブルファイル | +- インデックスファイル | +- フリースペースマップファイル | +- ビジビリティマップファイル +- WALファイル [*2] +- WALアーカイブ +- サーバログファイル [*1]基本的には1テーブルが1ファイルとなる ・1つのファイルに複数の表のデータは格納されない ・表領域（テーブルスペース）は任意のディレクトリに表データを格納したい場合に使用する [*2]先行書き込みログ（トランザクションログファイル）	OFA(Optimal Flexible Architecture)に準拠した構成で管理される ■OFA(Optimal Flexible Architecture)概要 所有するユーザおよび、バージョンの異なる複数のデータベースが共存できるように、データベース・ソフトウェアとデータベースを適切に配置および構成する 初期化パラメータファイル[*1] 制御ファイル [*2] データファイル [*3] REDOログファイル アーカイブログファイル [*1]インスタンス構成情報 [*2]データベース構成情報 [*3]表領域のデータ（Oracleが使用する「システム表領域」「UNDO表領域」「一時表領域」などの表領域、およびユーザが使用する表領域） ・通常1つの表領域は多数の小型ファイルで構成されるが、単一の大型ファイルで構成することも可能 ・1つのデータファイルに複数の表を格納することも、1つの表のデータを複数のデータファイルに分割して格納することもできる	インスタンス構成によってACLディレクトリが構成される ■スタンドアロン インスタンス 既定のインスタンスまたは、名前付きインスタンスが指定された既定構成 ■フェールオーバー クラスター インスタンス ソフトウェアに障害が発生した際、インスタンスを冗長ノードで保護する ■ファイル構成 プライマリ データ ファイル [*1][*3] セカンダリファイルグループ セカンダリ データ ファイル [*2][*3] トランザクション ログ ファイル [*1]システム情報（Oracleの制御ファイルに相当） [*2]テーブルなどのユーザデータ [*3]Azure Virtual Machine(VM)上で稼動している場合データファイルを表領域に複数割り当てた場合、自動的に64Kbyteごとに分散して格納される	ノード毎にデータベースファイル、表スペース コンテナーや、トランザクションログが管理される インスタンス構成情報 データベースマネージャ構成ファイル [*1] データベース構成情報 データベース構成ファイル [*2] トランザクションログ 自動ストレージパス 表スペースコンテナー [*3] など [*1]データベースマネージャ構成ファイルにインスタンス情報を格納（インスタンス内のデータベースに共通の情報として適用される） [*2]各データベース構成情報はデータベース構成ファイルに記述される [*3]システム情報を「カタログ表スペース」に、テーブルなどのユーザデータを「ユーザ表スペース」に格納。表スペースに対しコンテナ（ディレクトリ/ファイルで構成される物理オブジェクト）を複数配置することが可能。コンテナーはOracleのデータファイルに相当
4	インスタンスとデータベースの関係	データベースインスタンスとデータベースが1対1か、インスタンス内にデータベースを複数持てる構成か	1対N 唯一インスタンスがデータベースの集合である「データベースクラスタ」を管理 ※「データベースクラスタ」が他のDBMSで「インスタンス」と呼ばれるものに相当 ※インスタンス内に複数のデータベースを持つことができる	1対1 1つのインスタンスにつき、インスタンス構成情報とデータベース構成情報を1つずつ持つ	1対N インスタンスの構成情報を1つのシステムデータベースに格納し、各データベース構成情報はユーザデータベースのシステムテーブルに格納 ※1つのインスタンスに複数のデータベースを持つことができる	1対N データベースマネージャ構成ファイルにインスタンス情報（インスタンス内のデータベースに共通の情報）を格納する 各データベースの構成情報はデータベース構成ファイルに記述される ※1つのインスタンスに複数のデータベースを持つことができる
5	認証方式	DB接続ユーザとOSユーザとのリンクや認証方式	OSのユーザを使用する、データベース接続ユーザとOSのユーザを別にする、リンクさせる、が可能 ・trust認証 ・パスワード認証 ・GSSAPI認証 ・SSPI認証 ・Ident認証 ・Peer認証 ・LDAP認証 ・RADIUS認証 ・証明書認証 ・PAM認証	[ユーザ認証] ・OS認証 ・データベース認証 の2通り [ログインアカウント] インスタンスごとにログインアカウントを作成	[ユーザ認証] ・Windows／OS認証 ・SQL Server認証（データベース認証） の2通り [ログインアカウント] インスタンスとデータベースへの認証が別 ※インスタンスごとのログインアカウントを作成し、データベースごとのデータベースユーザを作成、データベースユーザをログインアカウントと関連付けする	[ユーザ認証] 外部機密保護サービス認証 （OS上のユーザーおよびグループの単位で認証） [ログインアカウント] 外部機密保護サービスに登録したユーザーに対し、管理者がデータベース接続権限を付与する
6	メモリ管理	DBサーバの各メモリ領域のサイズ管理のされ方	データベースクラスタのパラメータに基づいてメモリ確保される[*1][*2] 大きく分けて「共有メモリ」と「ローカル ヒープ」から構成される ■共有メモリ ・複数のバックエンドで共有されるデータを保持 ・セッションをまたいで共有すべきデータ ・共有バッファ、トランザクション情報、ロック情報　等 ・サービス起動時に固定サイズを確保 ■ローカルヒープ ・個別のセッションで使用するメモリ ・ソート、演算処理などを実行するときに使用するメモリスペース ・必要となるたびに確保、不要になったら開放 [*1]postgresql.conflにてキャッシュメモリサイズ、チェックポイント実施間隔、WALバッファサイズなどの項目を指定する [*2]メモリ確保/解放は「メモリコンテキスト」という単位で行われる	■手動メモリ管理 メモリー関連の初期化パラメータの設定にて各メモリコンポーネント毎に個別にサイズを指定する ■自動メモリ管理 [*1][*2] 有効にすると負荷に応じて運用中も自動的にサイズ調整、最適化される [*1]一部のメモリコンポーネントは自動チューニングの対象外 [*2]Oracle9i Database 以降で自動メモリ管理機能が強化されており、11g以降ではPGAおよびSGAの合計割り当てサイズを指定する形となっている	サーバー構成オプションにて最小メモリ、最大メモリ、最大ワーカスレッド数などを指定する。メモリ使用量はメモリマネージャー コンポーネントにより負荷状況に応じて自動的にサイズ調整、最適化される ■仮想アドレス空間(Virtual Address Space) [*1][*2] ・ユーザモード用（2GB） ・カーネルモード用（2GB） [*1]ユーザモード用および、カーネルモード用から構成され、4GBの領域が確保される。物理メモリのサイズが8GBであっても32bit環境では、常に4GBとなる [*2]仮想アドレス空間は、あくまでも「仮想」である為、仮想アドレス空間でのメモリ割り当てがそのまま物理メモリ上の領域を使用するわけではない	各メモリーセット[*1]について上限サイズ[*2]までメモリが確保されメモリーセット内の各用途のメモリリソースの間で動的に分配[*3]される [*1]OS から取得したメモリは「メモリー・セット」として管理されており、各コンポーネントは「メモリー・ブロック」という単位で必要なメモリを要求し「メモリー・マネージャー」が「メモリー・セット」から割り振る [*2]構成パラメータに上限値の指定項目があり、デフォルトは「AUTOMATIC」となっている [*3]「データベース・メモリー・セット」を例にすると、データベース活動化時に固定サイズが確保される領域（バッファプールなど）、クリーンアップされるまで使用量が単調増加する領域（カタログ・キャッシュ・ヒープなど）、都度使用量が増減する領域（共有ソート・ヒープなど）がある

【別紙】アーキテクチャ比較表

No.	アーキテクチャ	説明	PostgreSQL	Oracle	SQL Server	DB2
7	クエリ実行フロー	SQL文の実行フロー	1)SQLをパースツリーに変換 2)パースツリーを解析しクエリツリーに変換（アナライズ）[*1] 3)必要なクエリの書き換えを実施（リライト）[*2] 4)問合せを実行するプランツリーを作成（クエリオプティマイズ） 5)問合せを実行（エグゼキュート） [*1]テーブル名をOIDに変換する [*2]VIEWやRULEはリライトにより実装されている	1)解析処理(PARSE) ・ライブラリキャッシュチェック [*1] ・構文チェック [*2] ・表、列の定義チェック [*2] ・オブジェクト権限チェック [*2] ・実行計画作成 [*3] 2)実行処理(EXECUTE) [*4] 3)データ取り出し処理(FETCH) [*5] [*1]同一SQLの解析結果が共有プールのライブラリキャッシュ領域にキャッシュされている場合は、残りの解析処理はスキップ(SOFT PARSE) [*2]データディクショナリに対して再帰SQLが発行される(HARD PARSE) [*3]統計情報より最適な抽出方法を決定。実行計画は共有プールのライブラリキャッシュ領域にキャッシュする [*4]挿入、更新、削除処理はここまでで完了 [*5]SELECT文の場合のデータ取り出し。データがデータベースバッファキャッシュに存在する場合はキャッシュから取得する	1) コンパイル 1-1. ステートメントを解析 1-2. シーケンスツリーを作成 1-3 ツリーを正規化 1-4. ステートメント判定 1-4-1. SQL DMLの場合 1-4-1-1. TSQLステートメント プロシージャーとして、コンパイル 1-4-2. SQLDML以外の場合 1-4-2-1. シーケンスツリーをクエリグラフに変換 1-4-2-2. クエリグラフを最適化 1-4-2-3. クエリ実行プランを作成 2) 最適化 2-1. トリビアルなプラン最適化 2-2. 完全最適化 3) 問い合わせ実行 ストアドプロシージャのクエリプランだけでなく、アドホッククエリや自動パラメータ化クエリのプランもキャッシュに入れて、再利用できるようにする	・DB2 オプティマイザーにより、SQLステートメントから最適なアクセス・パスを解析 1) アクセス・パスにより解析処理 1-1. 索引アクセス[*1] 1-1-1. 索引参照 1-1-2.索引スキャン 1-1-2-1. マッチング索引スキャン 1-1-2-2. 非マッチング索引スキャン 1-1-3. 完全スキャン 1-2. 結合方式[*2] 1-2-1. ネスト・ループ結合 1-2-2. マージ・スキャン結合 1-2-3. ハッシュ結合 2)アクセス・パスからコンパイル 3)問い合わせを実行 [*1] SQLステートメントの最低限1つが索引付け可能であること [*2] 結合方式の場合、小さい表を選択し、結合対象の内部表への再アクセス回数を減らしている ※Visual Explain ツールを用いることでアクセス・パスを目視することが可能
8	アクセス・パス	表データのスキャン・結合方式	■表データの構成 データ・ブロックの連続で表データを構成 ■全表スキャン シングル・ブロック読み込み（1回の読み込み命令で1つのデータ・ブロックにアクセス） ■主キーを利用した一意索引スキャン 索引からTIDを参照し、TIDにより表データ・ブロックにアクセス ■非一意索引スキャン 一意索引スキャン同様 ■表結合 ネスティッド・ループ結合、ソート・マージ結合、ハッシュ結合を実装 ■オプティマイザ コストベースオプティマイザ [*1][*2] [*1]カラム間の相関関係などは考慮しない（複雑なクエリでは常に最適なプランが選択されるとは限らない） [*2]ヒント句機能はPostgreSQL拡張モジュールの「pg_hint_plan」により実装	■表データの構成 ・データ・ブロックの連続で表データを構成 ■全表スキャン ・マルチ・ブロック読み込み（複数のデータ・ブロックを一回の読み込みで処理し全表スキャン時の読み込み回数を軽減） ■主キーを利用した一意索引スキャン 索引からROWIDを参照し、ROWIDにより表データ・ブロックにアクセス ■非一意索引スキャン 一意索引スキャン同様 ■表結合 ネスティッド・ループ結合、ソート・マージ結合、ハッシュ結合を実装 ■オプティマイザ コストベースオプティマイザ [*1][*2] [*1]Oracle10gよりルールベースオプティマイザ(RBO)は動作保証外になり一本化された [*2]ヒント句により表の特定のアクセス・パスを使用するように指示可能	■表データの構成 ・基本ユニットは、ページ(page)で管理されており、8KB毎の連続データブロックで構成 ■オプティマイザー クエリ オプティマイザーでは、論理クエリ プランが作成されると、100以上から構成される物理操作プラン[*1]から最も効率的な物理操作を選択する [*1] 主な物理操作プラン 1. Clustered Index Delete WHERE 述語がある場合、その述語に適合する行だけが削除される 2. Index Scan 列に指定された非クラスター化インデックスからすべての行を取得する 3. Table Scan 列で指定されているテーブルからすべての行を取得する 4. Collapse 更新処理が最適化されます。クエリ プロセッサでは、同じキー値を削除して挿入する操作が隣接する行で検出されると、これらの個別の操作を 1 つのより効率的な更新操作に置き換える	■表データの構成 ページと呼ばれるブロックに格納され、ページのサイズは、4KB、8KB、16KB、32KB の4種類。データ・ページには、列の定義情報は含まれない ■表スペース・スキャン・アクセス 作成済みの一時表を介してアクセスされ、索引スキャンが不可能である場合に決定される ■ハッシュ・アクセス 検索条件式を使用した単一行にアクセスする照会で決定される ■集約関数アクセス SQLステートメントで集約関数を選択した際に決定される ■索引アクセス [*1] 索引を使用した 列・テーブル・ビューで指定される表にアクセスする ■INリスト・アクセス IN述部を含む照会に対して、INリスト索引スキャンまたは、INリスト直接表アクセスのいずれかが決定される ■直接行アクセス 条件に該当する行を見つめるのに、索引や表スペース・スキャンの使用を必要としない場合に決定される。きわめて高速なアクセス方式 [*1] 索引タイプ 特性は、「一般特性」と、「固有特性」の2つのカテゴリーに分けられる
9	同時実行性	読み取り一貫性を保ちながら同時実行性を実現するために採用している仕組み	MVCC（多版型同時実行制御） ・追記型アーキテクチャ（更新、削除の場合でも以前の行データは削除されず、ロールバックに必要な情報として残される） ・あるセッションの変更がコミットされていない時に、ほかのセッションが同じ情報を参照する場合、以前の行データを参照する	MVCC（多版型同時実行制御） ・DML（レコード操作）にて変更する前の情報はUNDO表領域内のUNDOセグメント領域に保存 ・あるセッションの変更がコミットされていない時に、ほかのセッションが同じ情報を参照する場合、UNDOセグメントに保存された情報（参照時に確定されている情報）だけを参照	ロック法 規定の設定ではロッキングメカニズム ■ペシミスティック同時実行制御 ユーザーが他のユーザーに影響するデータの変更を行うことを防止。ユーザがロックを解放するまで他のユーザはロックと競合する操作を実行できない ■オプティミスティック同時実行制御 データを読み取る時点ではロックされない。データを更新するときに、そのユーザーが読み取ってから他のユーザーによる変更がなかったかを確認。他のユーザがデータを更新していた場合、エラーが発生する ・多バージョン法のサポート SQL Server2005 からは、多バージョン法を使用した同時実行制御をサポート	ロック法 既定の設定ではロッキングメカニズム ・作業単位（コミット操作、フル・ロールバック操作、またはアプリケーション・プロセスの終了）毎に、ロックをかける ・排他ロックは、別のトランザクションの全てのロックをブロッキングする ・共有ロック同士であれば競合は発生しない。ロックに互換性がある ■ペシミスティック・ロッキング ・更新するデータの参照時に(U)ロックを取得 ・他アプリケーションは、参照できるが更新できない ■オプティミスティック・ロッキング ・参照から更新に間にロックを保持しない ・他アプリケーションは、参照・更新とも可能 ・参照から更新の間に、その行が他アプリケーションによって更新されていなければ、更新は成功する。その行が他アプリケーションによって、更新されていれば、更新は失敗する

【別紙】アーキテクチャ比較表

No.	アーキテクチャ	説明	PostgreSQL	Oracle	SQL Server	DB2
10	ロック機構	ロックの種類、リードで共有ロックを取得するかの違い、ロックエスカレーションの有無など	明示的ロック 　テーブルレベルロック 　行レベルロック 　ページレベルロック 　勧告的ロック ロックエスカレーション機構なし（行レベルのロック量が増えることによって、自動的にテーブルロックに変更されることはない） ■テーブルレベルロック テーブルに対するロック ■行レベルロック 同じ行に対する書き込みとロックだけをブロックする。異なる副トランザクション内であっても、同じ行に対して競合ロックを保持できる ■ページレベルロック ページレベルの共有/排他ロックがあり、これらは共有バッファプールにあるテーブルページへの読み書きのアクセスを管理するために使用される ■勧告的ロック 独自の意味を持つロックを生成する手法。勧告的ロックの使用に関してシステムによる制限はない	DMLロック DDLロック ■DMLロック 複数のユーザが同時にアクセスするデータの整合性を保障する。 ・行ロック(TX) ・表ロック(TM) ・行共有表ロック(RS) ・行排他表ロック(RX) ・共有表ロック(S) ・共有行排他行ロック(SRX) ・排他表ロック(X) リードロックを取得しない（更新を前提としたデータ読み取りの際はFOR UPDATE句にて明示的に行ロックを取得する必要がある） ロックエスカレーション機構なし ■DDLロック スキーマ・オブジェクトの変更や参照を実行する可能性のあるDDL操作によって干渉がおきないように「排他DDLロック」「共有DDLロック」を提供する	共有(S)ロック 更新(U)ロック 排他(X)ロック インテント ロック スキーマ ロック 一括更新(BU)ロック キー範囲 ・ロック単位として「ページロック」が存在する ・ロックエスカレーション機構あり ・行ロック、インデックスキー範囲ロックの際にそれらを含んでいるページのページロックを掛ける場合や、表ロックにエスカレートする場合がある ・READ_COMMITTED_SNAPSHOT オプションがOFFの場合、参照処理を行うと共有ロックを取得し、参照を完了すると、共有ロックが開放される	Sロック（共有） Uロック（更新） Xロック（排他） 各ロックは行または、ページロックが適用される ■ロックの保持/開放 ・作業単位が完了するまで、ロックが保持される ・分離レベルCSでは、カーソルを移動すると開放。分離レベルRSでは、選択行にロックが残る。分離レベルRRでは、走査行にロックが残る。カーソルをcloseしてもロックは外れない ・カーソルが行にある間リードロックを取得する（更新を前提としたデータ読み取りの場合はトランザクション分離レベルを設定して他のトランザクションからの更新を防ぐ必要がある） 挿入、更新、削除は分離レベルとは関係なく対象となる行のすべての排他ロックが取得される。ロックエスカレーション機構あり。ロックを管理するメモリ領域の設定を行い、ロックエスカレーションを発生させないようチューニングする
11	トランザクション機構	オートコミットの動作の違い、サポートしているトランザクション分離レベルなど	DML（レコード操作）は自動コミットされる。 DDL（テーブル操作）は自動コミットされない。 トランザクション途中でエラーを出すと、最後にCOMMITをしても、ROLLBACKしたのと同じ扱いとなる トランザクション分離レベルは3種類をサポート[*1] ・コミット読み取り(READ COMMITTED) ・繰り返し可能読み取り(REPEATABLE READ) [*2] ・直列可能(SERIALIZABLE) 規定の分離レベルは「コミット読み取り」 [*1]未コミット読み取り(READ UNCOMMITTED)も指定可能であるが、PostgreSQLではコミット読み取りとして扱われる [*2]PostgreSQLの実装ではファントムリードの可能性がない	DML（レコード操作）について自動でコミットはされない。 DDL（テーブル操作）は暗黙コミットされる。 トランザクション途中でエラーを出し最後にCOMMITを行うと、正常に実行できたDMLについては処理が確定する トランザクション分離レベルは以下をサポート ・コミット読み取り(Read Committed) ・シリアライズ（直列化）可能(Serializable) 規定の分離レベルは「コミット読み取り」	トランザクションの開始にはBEGIN TRANSACTION文を使う。DML（レコード操作）、DDL（テーブル操作）はトランザクションの一部として扱われる。（コミットする前であればロールバックすることができる） トランザクション分離レベルは全てサポート ・未コミット読み取り(READ UNCOMMITTED) ・コミット読み取り(READ COMMITTED) ・繰り返し可能読み取り(REPEATABLE READ) ・直列可能(SERIALIZABLE) 規定の分離レベルは「コミット読み取り」 トランザクションは、暗黙的ではない。SQL文が実行されるたびに自動的にコミットされることを想定している	BEGIN文を使わずとも自動的にトランザクションが開始する。DML（レコード操作）、DDL（テーブル操作）はトランザクションの一部として扱われる。（コミットする前であればロールバックすることができる） 4つの分離レベルをサポート ・反復可能読み取り (RR) [*1] ・読み取り固定 (RS) [*2] ・カーソル固定 (CS) [*3] ・非コミット読み取り (UR) [*4] 規定の分離レベルは「カーソル固定」 [*1]ISO分離レベルのSerializableに相当 [*2]ISO分離レベルのRepeatable Readに相当 [*3]ISO分離レベルのRead Committedに相当 [*4]ISO分離レベルのUncommitted Readに相当
12	デッドロック検知	デッドロック検出の仕組み	ロック待ち検出→状態検査による検知 パラメータで指定した時間を超えてトランザクションがロック待ちしている場合、デッドロックの可能性があると判断してロック状態を検査する。デッドロックが検出されれば、1つのトランザクションを強制的にロールバックする	状況検知 デッドロック状況を自動的に検出し、デッドロックに含まれる文の1つをロールバックして競合する一連の行ロックを解放し、デッドロックを解決する。文レベルのロールバックの対象となったトランザクションにメッセージを通知する	定期間隔で検知 ロックモニタースレッドにより定期間隔毎に検出を行う[*1]。対象のトランザクションをロールバックして、アプリケーションにエラーを返す。既定ではロールバックに最もコストのかからないトランザクションを実行しているセッションがデッドロックの対象として選択される [*1]既定の間隔は5秒で、デッドロックが検出されると短い時間になり、検出されなくなると5秒に引き上げられる	定期間隔で検知 データベース構成パラメータに指定した間隔で定期検出を行う。デッドロックプロセスを任意に選択し、エラーを返却して非コミット・トランザクションを自動的にロールバックする
13	データ・ディクショナリ	メタデータ（データベース内のオブジェクト、表領域、ユーザ、権限などの情報）へのアクセスがどのように提供されているか	・「情報スキーマ」ビュー [*1] ・システムカタログ（表およびビュー） [*2] [*1]標準SQLで定義されたビュー [*2]PostgreSQL固有の機能についての情報はシステムカタログから取得する必要がある	「データ・ディクショナリ・ビュー」を介してOracleが管理するディクショナリ表の情報を取得・利用する	「システム ビュー」の「カタログ ビュー」を介してカタログ メタデータの情報を取得・利用する	「システム・カタログ・ビュー」が設けられている。 （SYSCATビュー および SYSSTATビュー） また、Oracle データ・ディクショナリー互換ビューをサポートしている
14	動的パフォーマンス情報の取得	システムパラメータ、メモリ使用量/割当て、統計情報などへのアクセスがどのように提供されているか	・統計情報ビュー ・統計情報アクセス関数	動的パフォーマンス・ビュー	・動的管理ビュー ・動的管理関数	スナップショット・モニター SQL 管理ビュー(SYSIBMADM スキーマ)
15	I/O分散	データを複数の物理ファイルに分散配置する仕組み	テーブルスペース機能を用いて表データファイルを別の物理ディスク上に配置することは可能 パーティショニング機能についてはpgpool-II等の別の機構を併用する必要がある	パーティション機能を用いてデータファイルを複数の物理ディスクに分散配置する	表領域に複数のデータファイルを割り当てし、分散配置されたファイルを複数の物理ディスクに配置する	表スペース（表領域）に対して複数の「コンテナ」を配置し、複数の物理ディスクに分散する